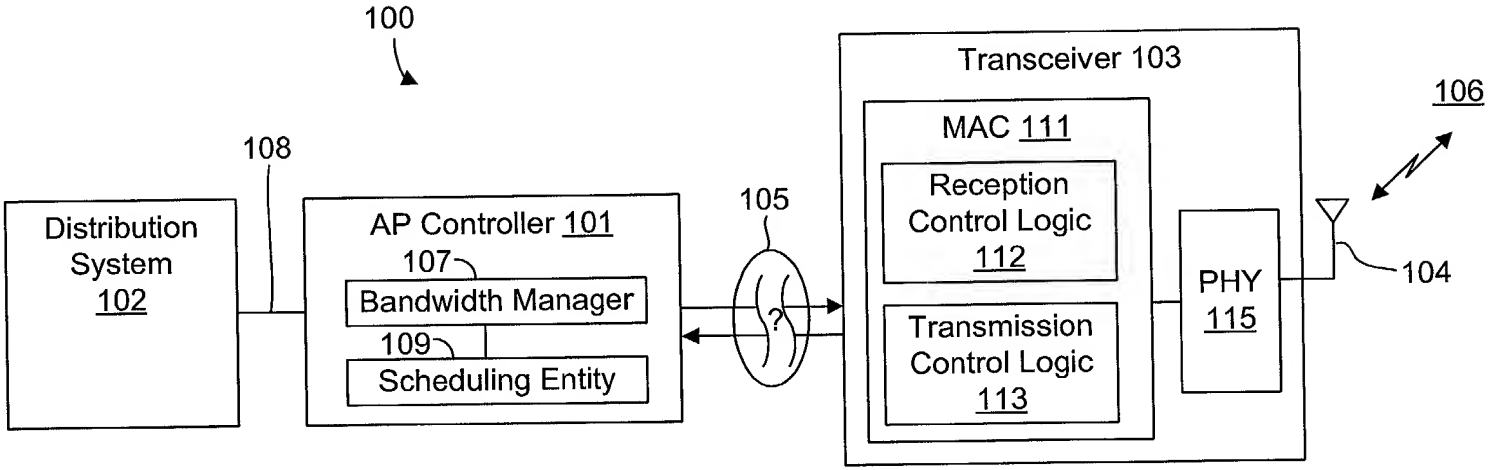

100
Transceiver 103
106
MAC 111
108
Reception
Control Logic
112
105
Distribution
System
102
AP Controller 101
107
Bandwidth Manager
PHY
115
104
109
Scheduling Entity
Transmission
Control Logic
113

FIG. 1

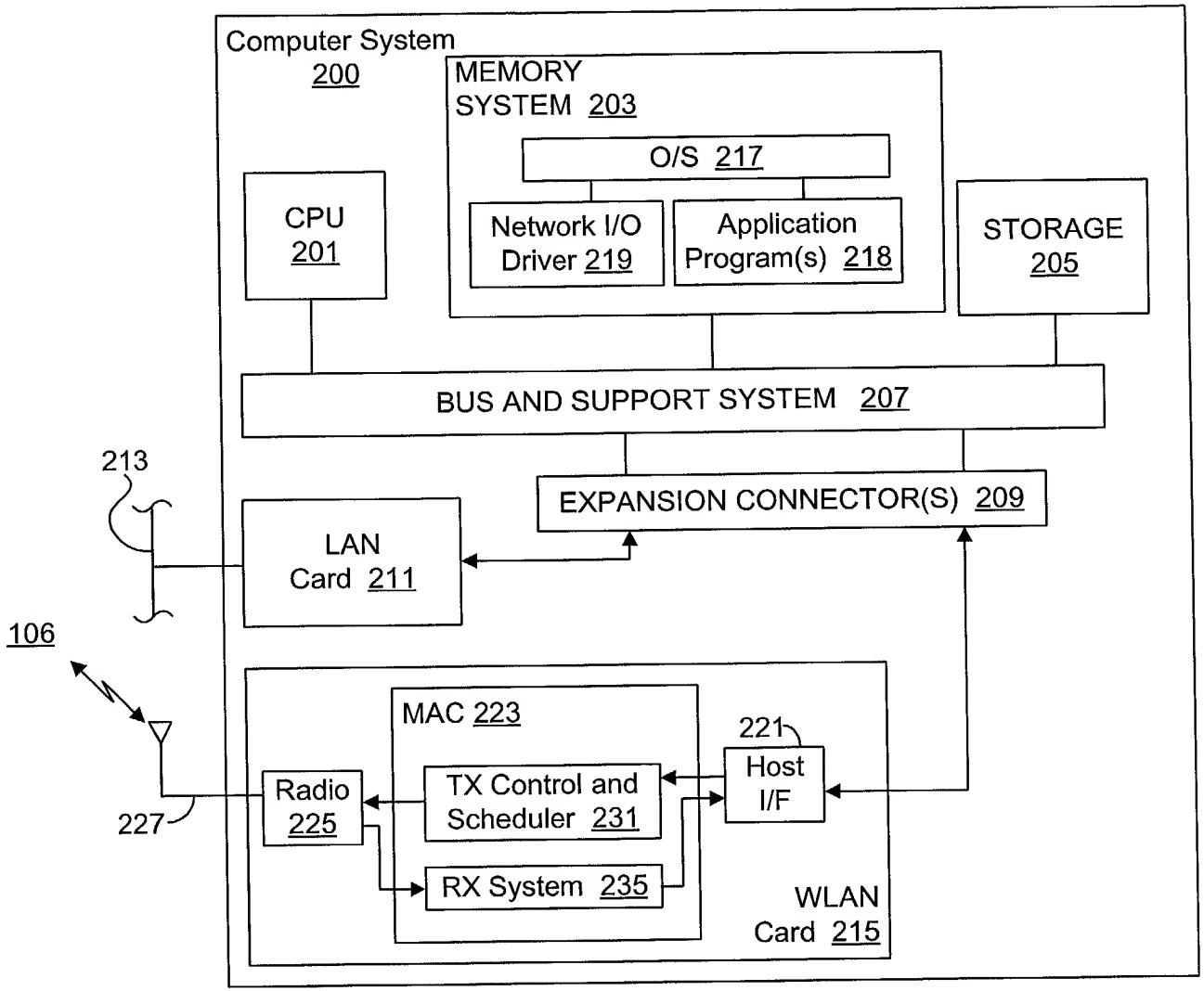
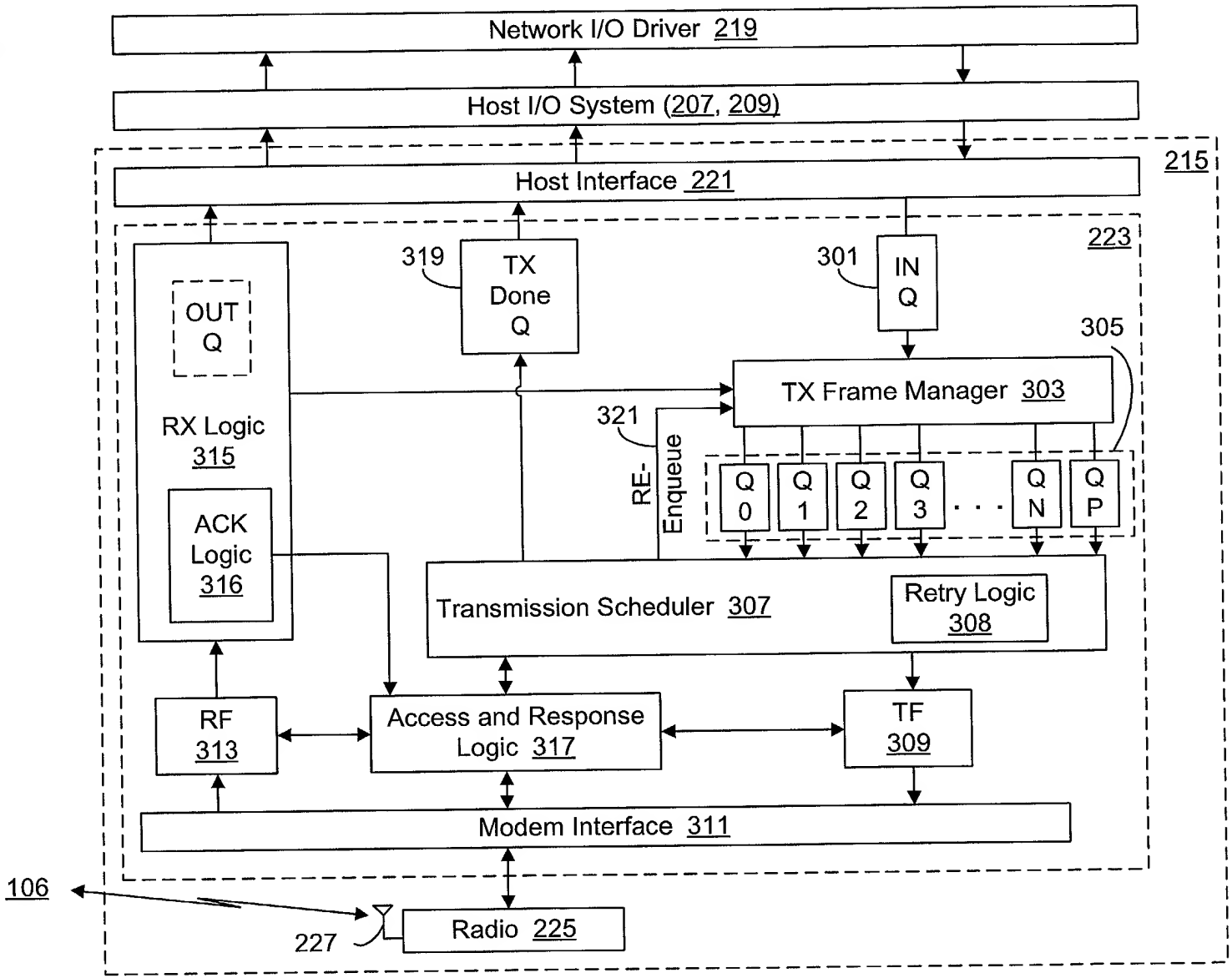

Computer System
200
MEMORY
SYSTEM 203
O/S 217
CPU
201
Network I/O
Driver 219
Application
Program(s) 218
STORAGE
205
BUS AND SUPPORT SYSTEM 207
213
EXPANSION CONNECTOR(S) 209
LAN
Card 211
106
MAC 223
221
Host
I/F
Radio
225
TX Control and
Scheduler 231
227
RX System 235
WLAN
Card 215

FIG. 2

FIG. 3


Network I/O Driver 219
Host I/O System (207, 209)
215
Host Interface 221
223
319
TX Done Q
301
IN Q
305
OUT Q
TX Frame Manager 303
321
RE-Enqueue
RX Logic 315
Q 0
Q 1
Q 2
Q 3
Q N
Q P
ACK Logic 316
Transmission Scheduler 307
Retry Logic 308
RF 313
Access and Response Logic 317
TF 309
Modem Interface 311
106
227
Radio 225

Tohoso' esoheo

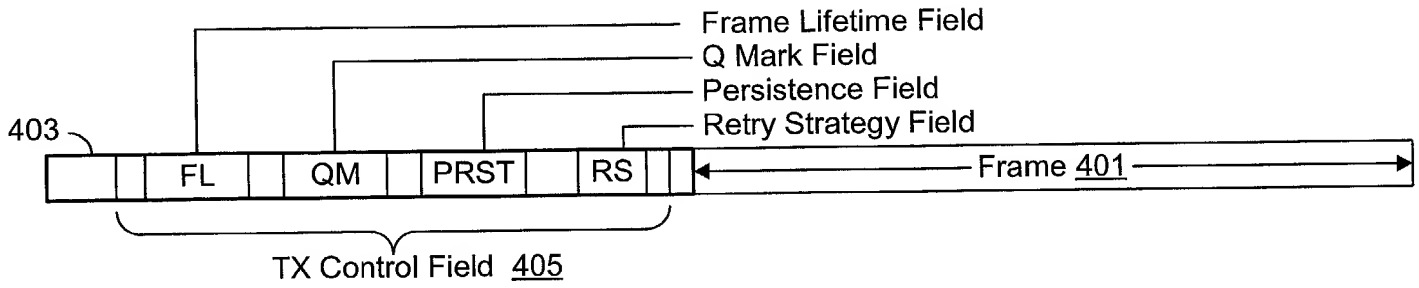


FIG. 4

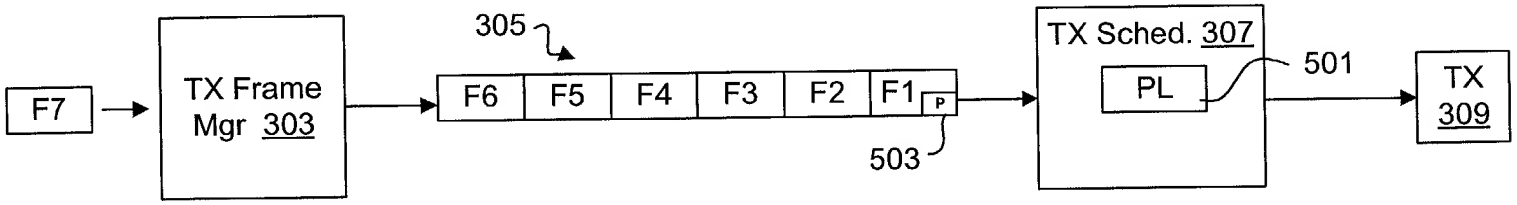


FIG. 5A

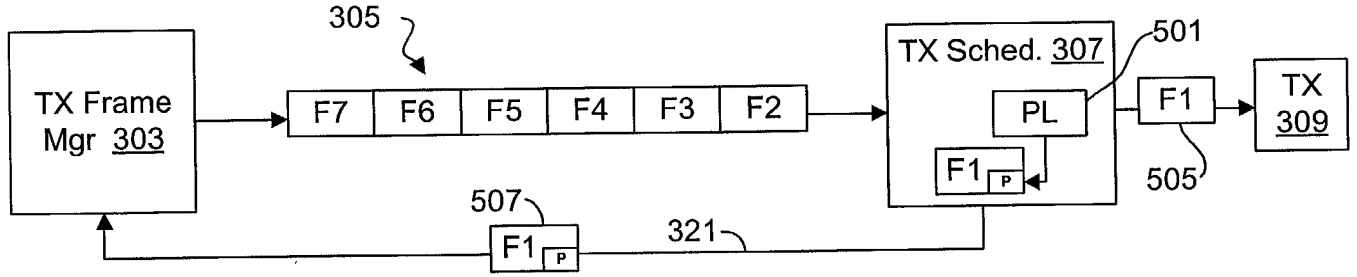


FIG. 5B

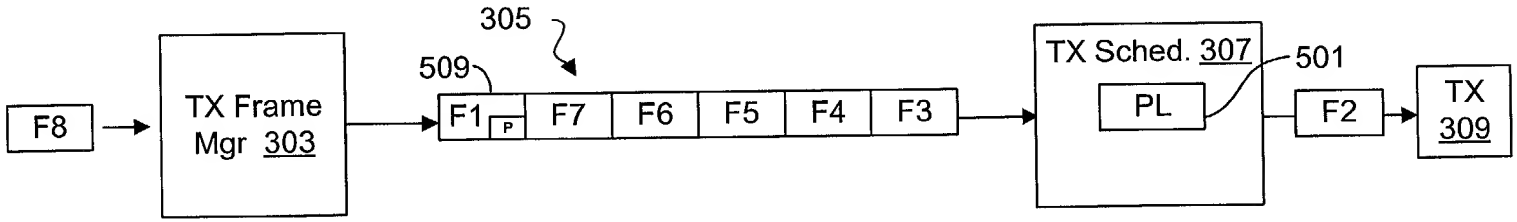


FIG. 5C

Patent Application

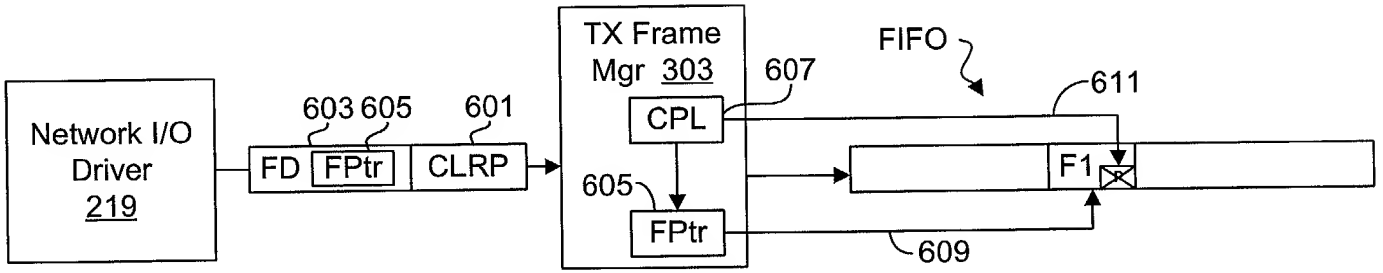


FIG. 6

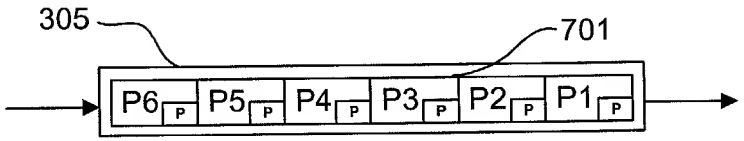


FIG. 7A

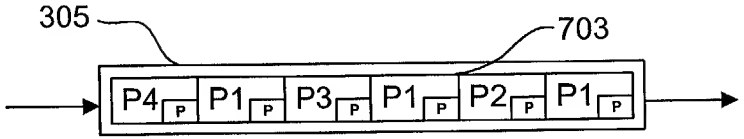


FIG. 7B

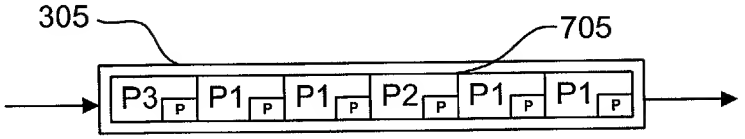


FIG. 7C

Tahoe-2506360

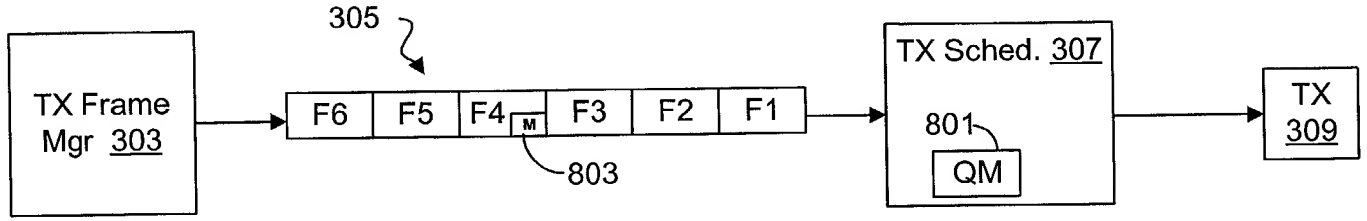


FIG. 8A

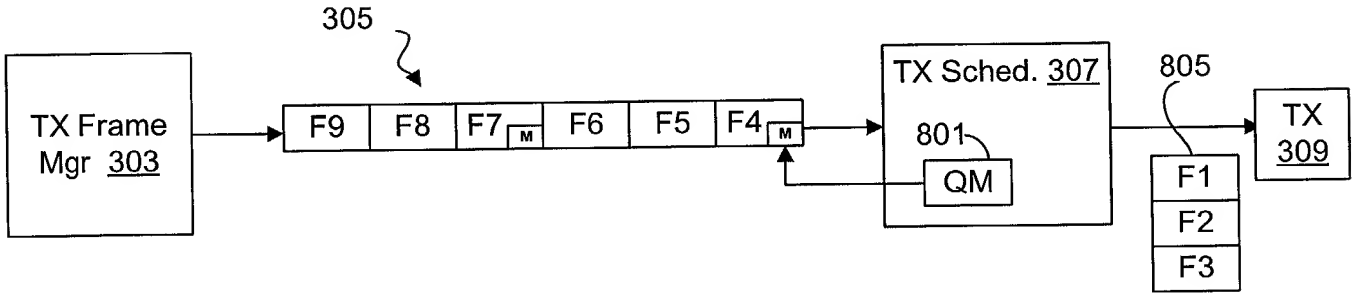


FIG. 8B

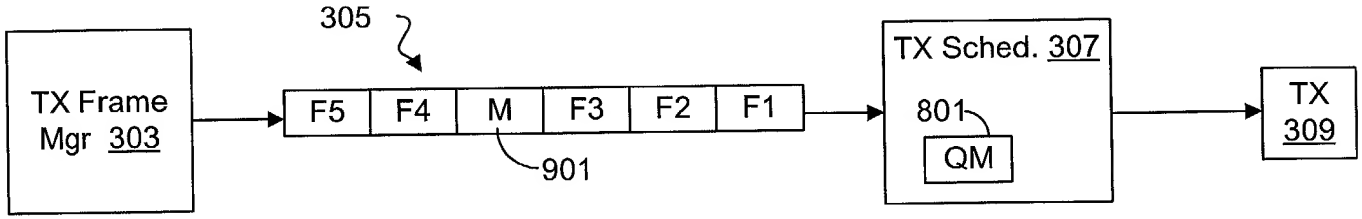


FIG. 9A

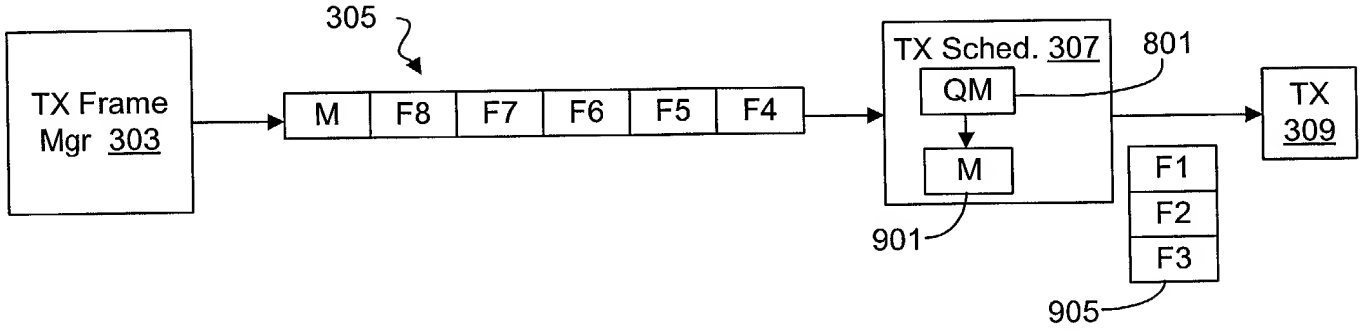


FIG. 9B

Patent Pending

FIG. 10

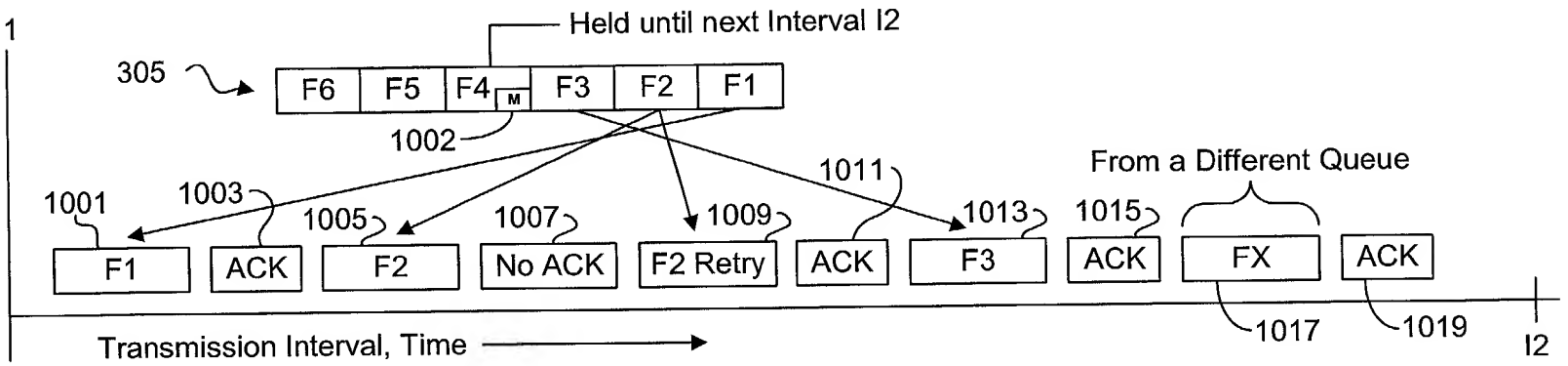
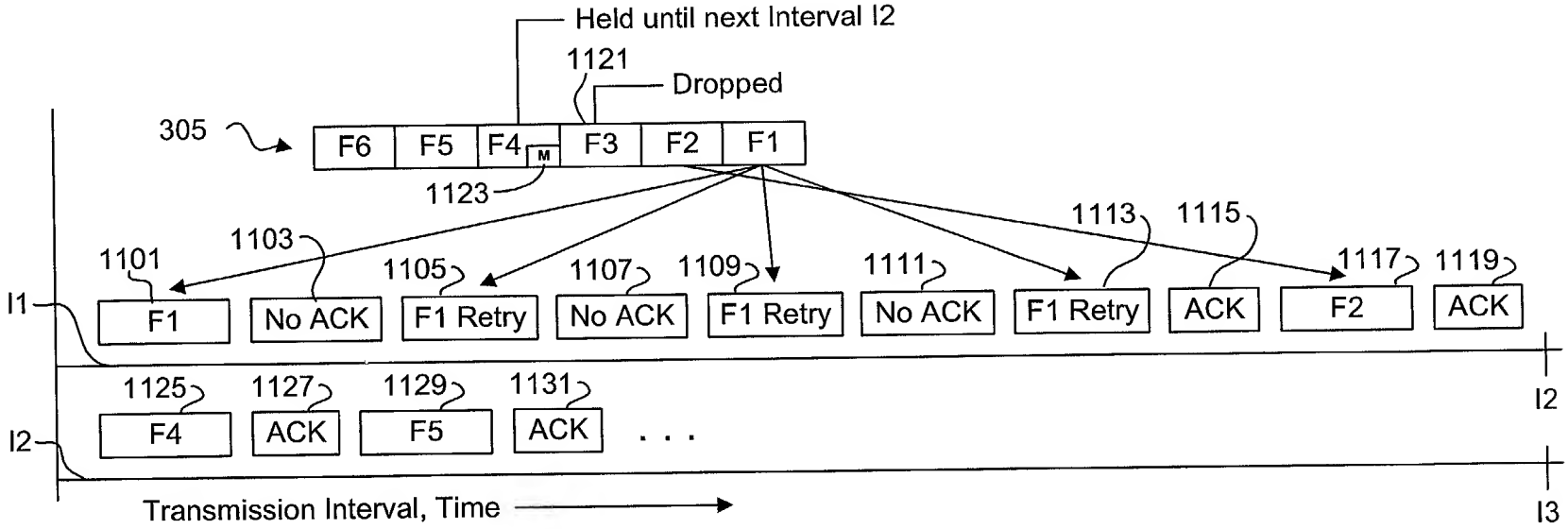


FIG. 11



Patent Application

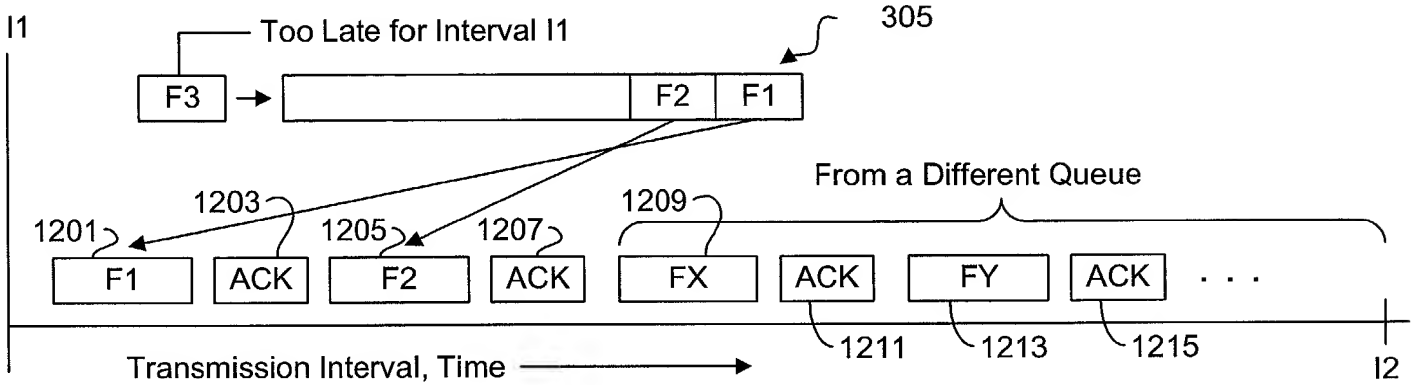


FIG. 12A

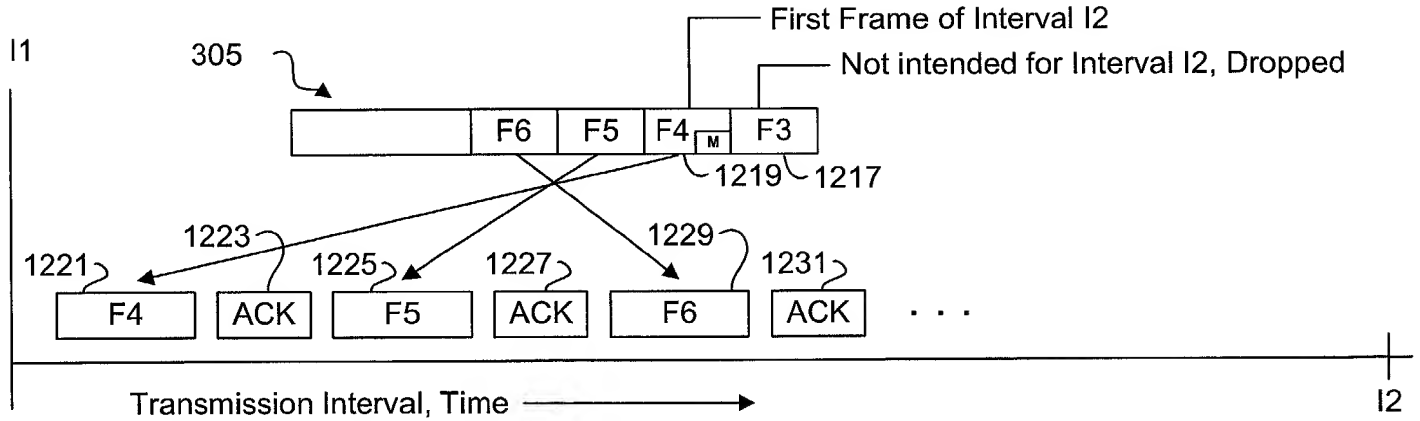


FIG. 12B

Potosa" esoeiseo

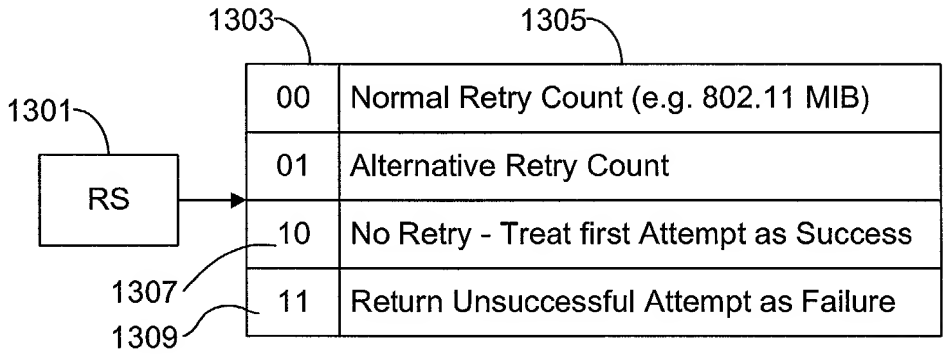


FIG. 13

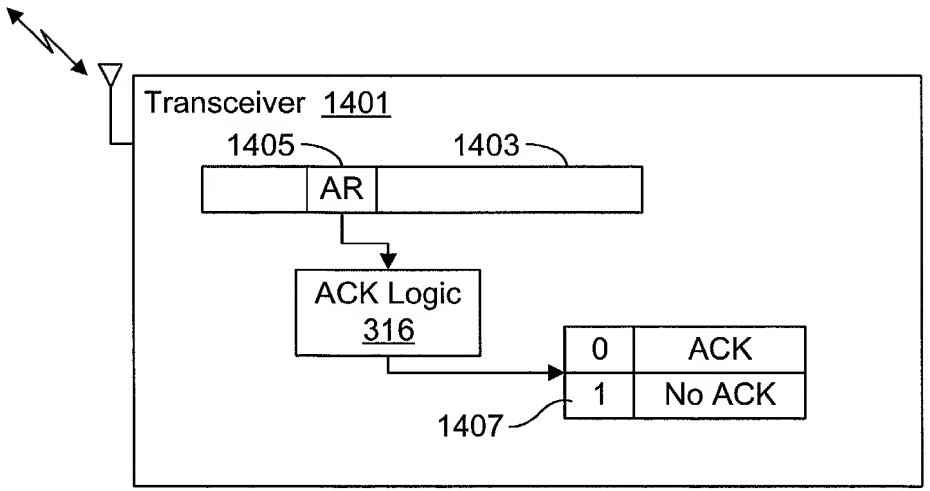


FIG. 14

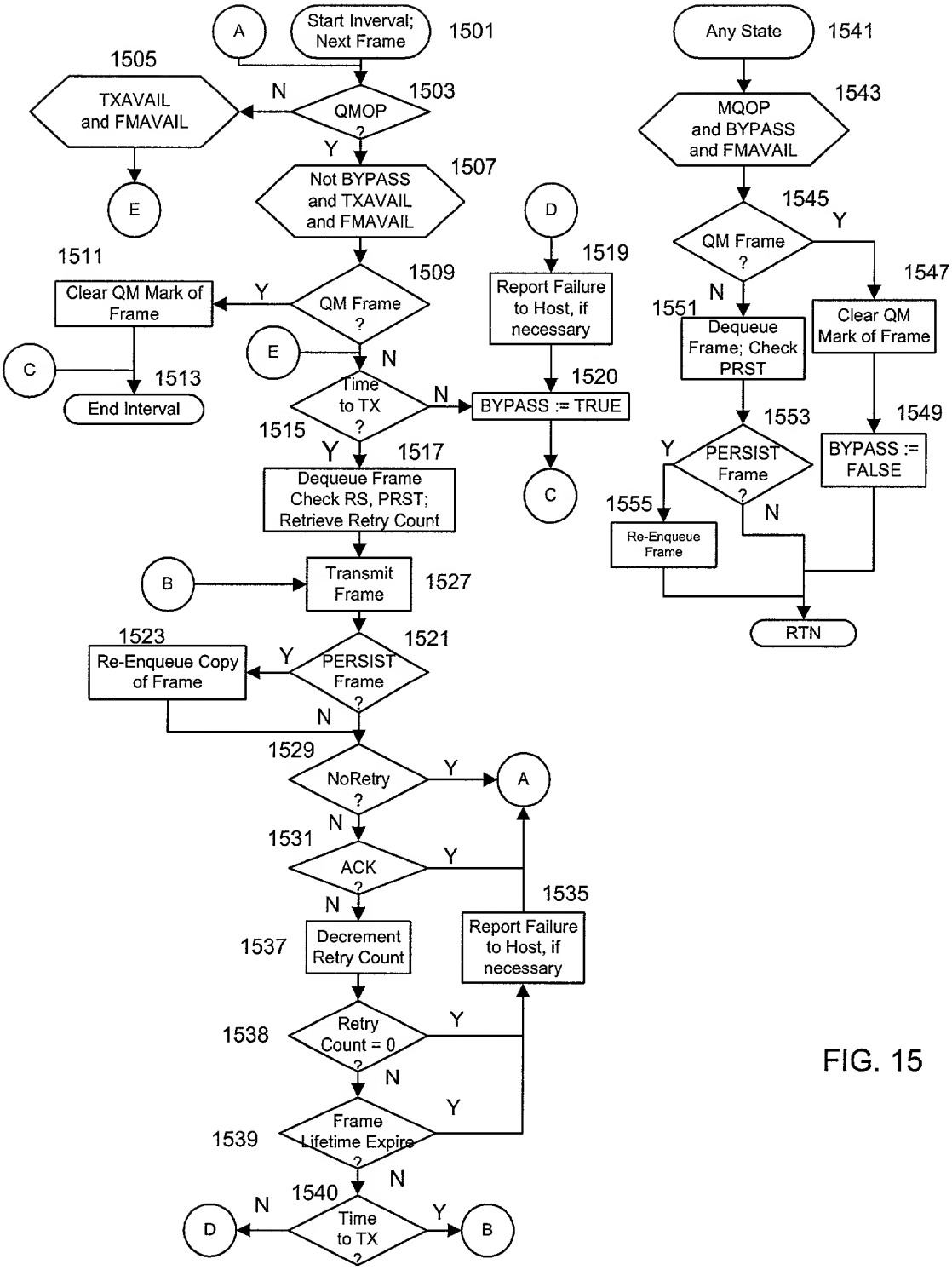

A
Start Inverval; Next Frame
1501
1505
TXAVAIL and FMAVAIL
N
QMOP ?
1503
Y
1507
Not BYPASS and TXAVAIL and FMAVAIL
E
D
1519
Report Failure to Host, if necessary
1520
BYPASS := TRUE
C
1511
Clear QM Mark of Frame
Y
QM Frame ?
1509
E
N
C
1513
End Interval
Time to TX ?
N
1515
Y
1517
Dequeue Frame Check RS, PRST; Retrieve Retry Count
B
Transmit Frame
1527
1523
Re-Enqueue Copy of Frame
Y
PERSIST Frame ?
1521
N
1529
NoRetry ?
Y
A
N
1531
ACK ?
Y
N
1535
Report Failure to Host, if necessary
1537
Decrement Retry Count
1538
Retry Count = 0 ?
Y
N
1539
Frame Lifetime Expire ?
Y
N
1540
D
N
Time to TX ?
Y
B
Any State
1541
1543
MQOP and BYPASS and FMAVAIL
1545
QM Frame ?
Y
1547
N
1551
Dequeue Frame; Check PRST
Clear QM Mark of Frame
1553
1549
Y
PERSIST Frame ?
BYPASS := FALSE
1555
N
Re-Enqueue Frame
RTN

FIG. 15

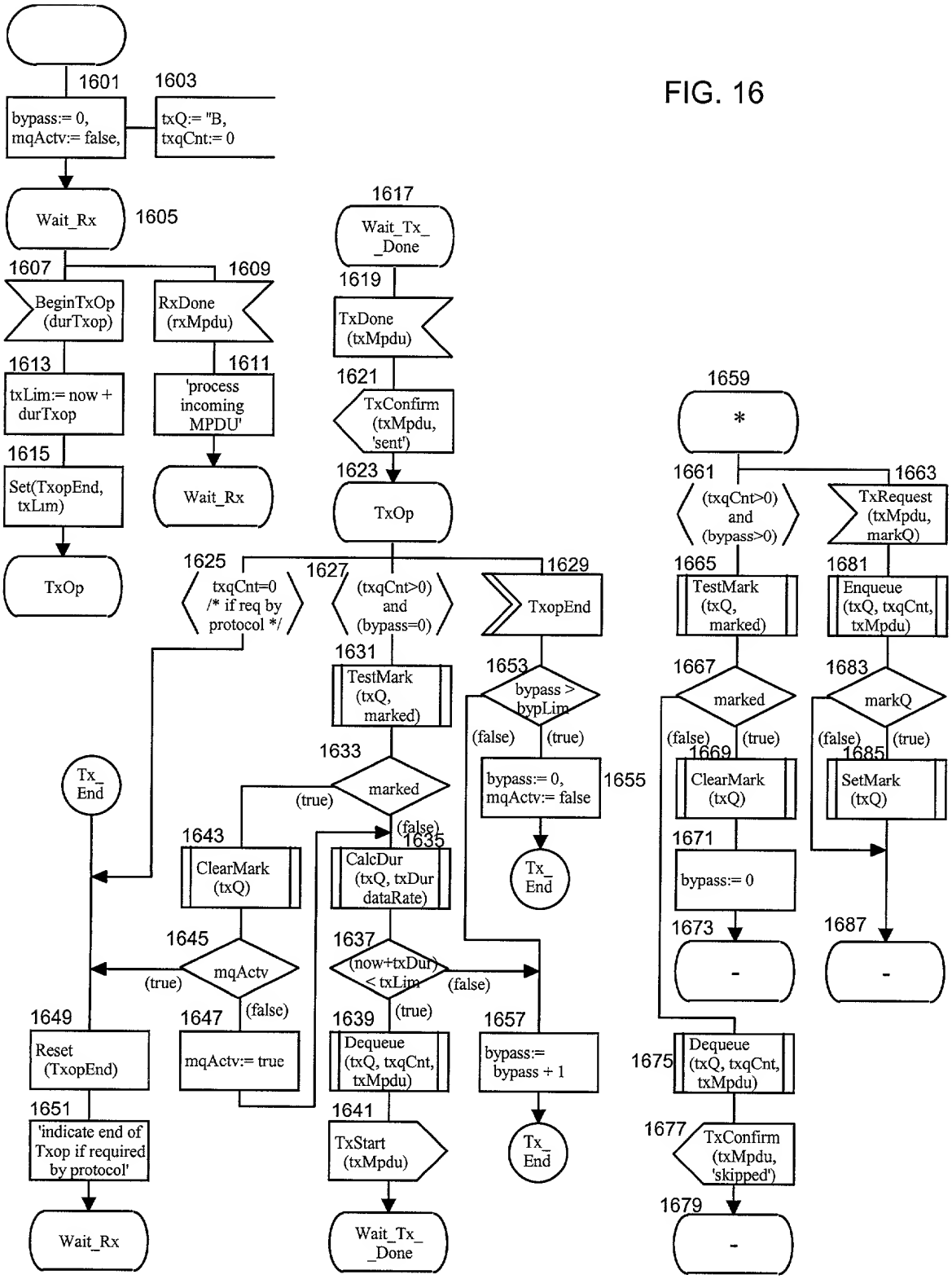

1601
1603
bypass:= 0,
mqActv:= false,
txQ:= "B,
txqCnt:= 0
Wait_Rx
1605
1607
1609
BeginTxOp
(durTxop)
RxDone
(rxMpdu)
1613
1611
txLim:= now +
durTxop
'process
incoming
MPDU'
1615
Set(TxopEnd,
txLim)
Wait_Rx
TxOp
1617
Wait_Tx_
_Done
1619
TxDone
(txMpdu)
1621
TxConfirm
(txMpdu,
'sent')
1623
TxOp
1625
txqCnt=0
/* if req by
protocol */
1627
(txqCnt>0)
and
(bypass=0)
1629
TxopEnd
1631
TestMark
(txQ,
marked)
1653
bypass >
bypLim
(false)
(true)
1633
marked
(true)
(false)
bypass:= 0,
mqActv:= false
1655
Tx_
End
Tx_
End
1643
ClearMark
(txQ)
1635
CalcDur
(txQ, txDur
dataRate)
1645
mqActv
(true)
(false)
1637
(now+txDur)
< txLim
(false)
(true)
1649
1647
1639
1657
Reset
(TxopEnd)
mqActv:= true
Dequeue
(txQ, txqCnt,
txMpdu)
bypass:=
bypass + 1
1651
1641
'indicate end of
Txop if required
by protocol'
TxStart
(txMpdu)
Tx_
End
Wait_Rx
Wait_Tx_
_Done
1659
*
1661
1663
(txqCnt>0)
and
(bypass>0)
TxRequest
(txMpdu,
markQ)
1665
1681
TestMark
(txQ,
marked)
Enqueue
(txQ, txqCnt,
txMpdu)
1667
1683
marked
markQ
(false)
(true)
(false)
(true)
1669
1685
ClearMark
(txQ)
SetMark
(txQ)
1671
bypass:= 0
1673
1687
-
-
1675
Dequeue
(txQ, txqCnt,
txMpdu)
1677
TxConfirm
(txMpdu,
'skipped')
1679
-

FIG. 16